Fachhochschule Wolfenbüttel

University of Applied Sciences

Electronic engineering faculty

Redesign of the STEP 7 Software controlling the sorting plant in the PLC Lab (Industrielle Steuerungen)

Submitted by:

Ismael Holgueras de Lucas

Supervising tutor:

Prof. Dr.-Ing. Michael Haas

February 2012

INDEX OF THE PICTURES

INDEX OF THE TABLES

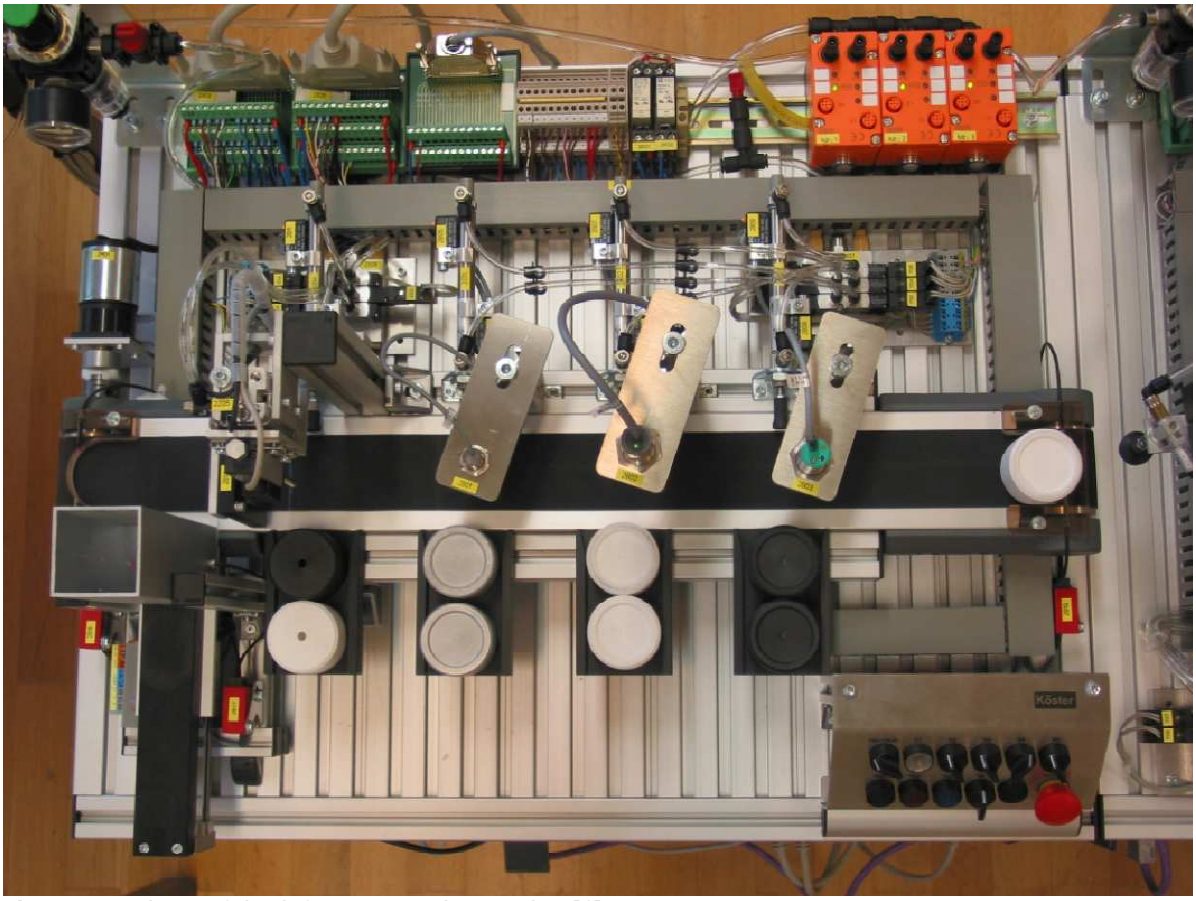
Chapter 1: Introduction

1.1 Goals of the project

The purpose of this final project is to redesign an existing program. It improves the old program with a new internal software structure. It consists in to program a Siemens PLC, with the software Step 7 of the same company. For this objective, I have used a plant that simulates a real processing plant. The aim of this plant is that students can develop the skills to program these automatons. The use of these systems is very wide in real industry.

1.2 Description of the plant

This plant consists of two parts, the first one is a belt in which are located four sensors that discriminate objects depending on their characteristics.



Picture 1: *Photo of the left part. "Sortierstrecke"* [1]

At the beginning of the belt, there is the feeder. It stores the pieces and places them in the belt. For this, it has rectangular prism where the pieces are stored; and a piece detector inside. The charger cylinder puts them in the belt.

The first sensor is a height measure sensor. It gets down and touches the pieces. The height value is stored in a memory variable. All of this sensors have a chute and a cylinder, this puts out the pieces that don´t fit the conditions. The maximum capacity of

a chute is two pieces. The condition of the height measure sensor is that the pieces are in the correct position. Each piece has two different sides, one with a hole and other with a rim. The last one is the correct position. Before this sensor, there is a photoelectric sensor that indicates when the piece is in the correct position for be measured.

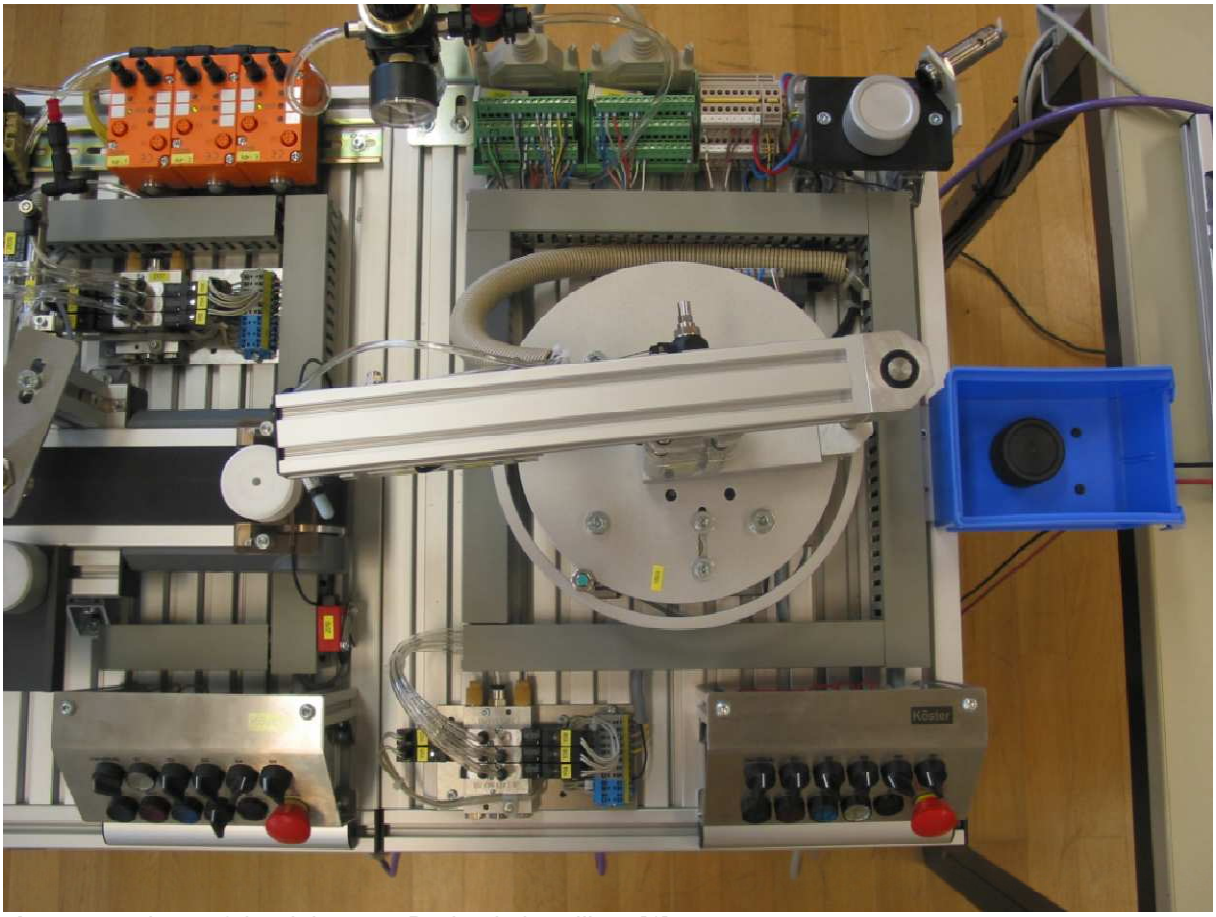
The second is a detector of metal pieces. It's an inductor sensor that detects only metal pieces.

The third has to detect the white pieces, it´s an optic sensor and detects the white and metal pieces. Owing to the black pieces don´t reflect the light of the sensor.

The last one detects the black pieces, it´s a capacitive sensor. So this sensor is activated by all items.

The conveyor belt has a sensor in the end of the belt for stop the belt and the next stage knows that can take the piece.

Also, this part has a button panel that allows the operator to control the plant. It has buttons, switches, lights and an emergency stop button.



Picture 2: *Photo of the right part. "*Drehachshandling*"* [1]

The right part consists in a robotic arm that takes the pieces of the belt´s end. And it puts them in the blue container.

The robot is formed of a circular table, this turns and sites the arm in the needed position. The arm has two cylinders that permit the arm to be allocated in four different positions: up extended, down extended, up retracted and down retracted. The arm has to be in the down extended position for take and drop a piece. But it can´t be in this position for turn, because the arm would strike the pressure valve. The plant has a security system with an emergency sound so that the programmer can´t damage the plant.

At the top left of the photo there is a black table with the same three sensor that in the left part. But it isn't used. This part also has a button panel for control it.

1.3 List of input and output signals

The next tables contain all the signals that the plant has. The first table is to the 'Sortierstrecke'.

Address		Description
I	0.0	Metal sensor
I	0.1	White sensor
I	0.2	Black sensor
I	0.3	Photoelectric sensor left
I	0.4	Photoelectric sensor right
I	0.5	Out limit switch of the metal cylinder
I	0.6	In limit switch of the metal cylinder
I	0.7	Out limit switch of the white cylinder
I	1.0	In limit switch of the white cylinder
I	1.1	Out limit switch of the black cylinder
I	1.2	In limit switch of the black cylinder
I	1.3	Out limit switch of the height cylinder
I	1.4	In limit switch of the height cylinder
I	1.5	Up limit switch of the height cylinder
I	1.6	Down limit switch of the height cylinder
I	1.7	Photoelectric sensor before height sensor
I	2.0	Out limit switch of the charger cylinder
I	2.1	In limit switch of the charger cylinder
I	2.2	Charger sensor
I	2.3	Switch S6 position 1
I	2.4	Switch S3
I	2.5	Switch S4
I	2.6	Switch S5
I	2.7	Switch S2
I	3.0	Start button
I	3.1	Stop button
I	3.2	Quit button
I	3.3	Switch S6 position 2
I	3.4	Switch S1
I	3.5	Emergency button

I	3.6	Switch manual-automatic position manual
I	3.7	Switch manual-automatic position automatic
Q	8.0	Metal cylinder out
Q	8.1	Metal cylinder in
Q	8.2	White cylinder out
Q	8.3	White cylinder in
Q	8.4	Black cylinder out
Q	8.5	Black cylinder in
Q	8.6	Height cylinder out
Q	8.7	Height cylinder in
Q	9.0	Sensor height
Q	9.1	Belt clockwise
Q	9.2	Start light
Q	9.3	Stop light
Q	9.4	Quit light
Q	9.6	Charger cylinder
Q	9.7	Belt anticlockwise

Table 1: *Signals of the left part. "Sortierstrecke"* [1]

Table of the input and output of the right part of the plant, 'Drehachshandling'.

Address		Description
I	4.0	Emergency button
I	4.1	Switch manual-automatic position manual
I	4.2	Switch manual-automatic position automatic
I	4.3	Switch S1 position 1
I	4.4	Switch S1 position 2
I	4.5	Switch S2
I	4.6	Switch S3 position 1
I	4.7	Switch S3 position 2
I	5.0	Switch S4 position 1
I	5.1	Switch S4 position 2
I	5.2	Switch S5
I	5.5	Quit button
I	5.6	Stop button
I	5.7	Start button
I	6.1	Left sensor of the arm
I	6.2	Middle sensor of the arm
I	6.3	Right sensor of the arm
I	6.4	Down limit switch of the arm
I	6.5	Up limit switch of the arm
I	7.0	Out limit switch of the arm
I	7.1	In limit switch of the arm
Q	12.2	Arm cylinder down

Q	12.3	Arm cylinder out
Q	12.4	Arm cylinder in
Q	12.5	Arm takes a piece
Q	12.6	Arm drops a piece
Q	13.0	Arm turn to the left
Q	13.1	Arm turn to the right

Table 2: Signals *of the right part "Drehachshandling"* [1]

1.4 Description of the control system

Each part of the plant is controlled by a S7-300 PLC. The PLC that controls the left side is composed of the following racks:

-Place 1: S7 - Power Supply.
-Place 2 and 3: CPU with integrated PROFIBUS Adapter.
-Place 4: Digital inputs, 16 inputs of 24 Volts DC.
-Place 5: Digital inputs, 16 inputs of 24 Volts DC.
-Place 6: Digital outputs, 16 inputs of 24 Volts DC and 0.5 A.
-Place 7: Digital outputs, 16 inputs of 24 Volts DC and 0.5 A.
-Place 8: Network adapter for Industrial Ethernet.

Rack 2:

-Place 1: S7 - Power Supply.
-Place 2 and 3: Profibus node.
-Place 4: 4 analogical inputs, 2 analogical outputs of 8 bits.
-Place 5: SIMATIC NET CP CARD.

The structure of the second PLC is as follow:

Rack 1:

-Place 1: S7 - Power Supply.
-Place 2 and 3: CPU with integrated Profibus Adapter.
-Place 4: Digital inputs, 16 inputs of 24 Volts DC.
-Place 5: Digital inputs, 16 inputs of 24 Volts DC.
-Place 6: Digital outputs, 16 inputs of 24 Volts DC and 0.5 A.
-Place 7: Digital outputs, 16 inputs of 24 Volts DC and 0.5 A.
-Place 8: Network adapter for Industrial Ethernet.

The communication between the PLCs and the computer can be done through the Ethernet wire or the USB to Profibus adapter. The Profibus network is used by the PLC to communicate.

Chapter 2: Methods

2.1 Configuration of the hardware

For the configuration of the automatons, I have followed the steps in the manual of the subject 'Labor Industrielle Steuerungen' [1]. Hardware configuration is contained on pages 3 and 13 of that manual.

2.2 Design process

The design process is the next:

First, I must analyze the requirements to solve the problem. What is the movement of the different pieces? What happens in each situation? One of the most important things is to know what sensors and actuators the system has.

The second is the definition phase in which it will be defined a solution to the problem. I have to develop a Moore state machine. Where the outputs are determined only by the current state and they don't dependent of the inputs.

In the third step, I have to transcribe the Moore machine in a language that it can use by Step7. These are different languages but I use the FUP for this project because it is more visual.

The last thing is to test the program in the real plant. It checks the real operation of the program.
This is an iterative process. When an error is detected, you have to go to a previous step and change what has been done.

2.3 Performance requirements

The functioning of the left part is the next:

I have chosen as criterion that only one piece at a time should be on the belt. When the start button may be activated, it starts loading items from the store, if it has.
The first station, through which it passes, is the measurement of height. To this end, a rod is lowered to measure if the height is correct. If so, send the piece to the chute of this station. The maximum capacity of this chute is two pieces. If it arrives more parts that the conditions let, it will have to send at the end of the belt. The criterion is that these parts cannot be used by the other stations, so these have to go directly to the end.

The second is one that matches the pieces detect when a piece is detected with these characteristics is sent to your chute. As in the rest of the stations if more than two pieces in the chute ship until the end without it being intercepted by the rest of stations.
The second station is one that detects the metallic parts. When a piece is detected with these characteristics is sent to its chute. As in the rest of the stations, if there are more than two pieces in the chute, it will send until the end and it won't be intercepted by the rest of stations.

The third station detects the white pieces and the operation is the same as shown above.

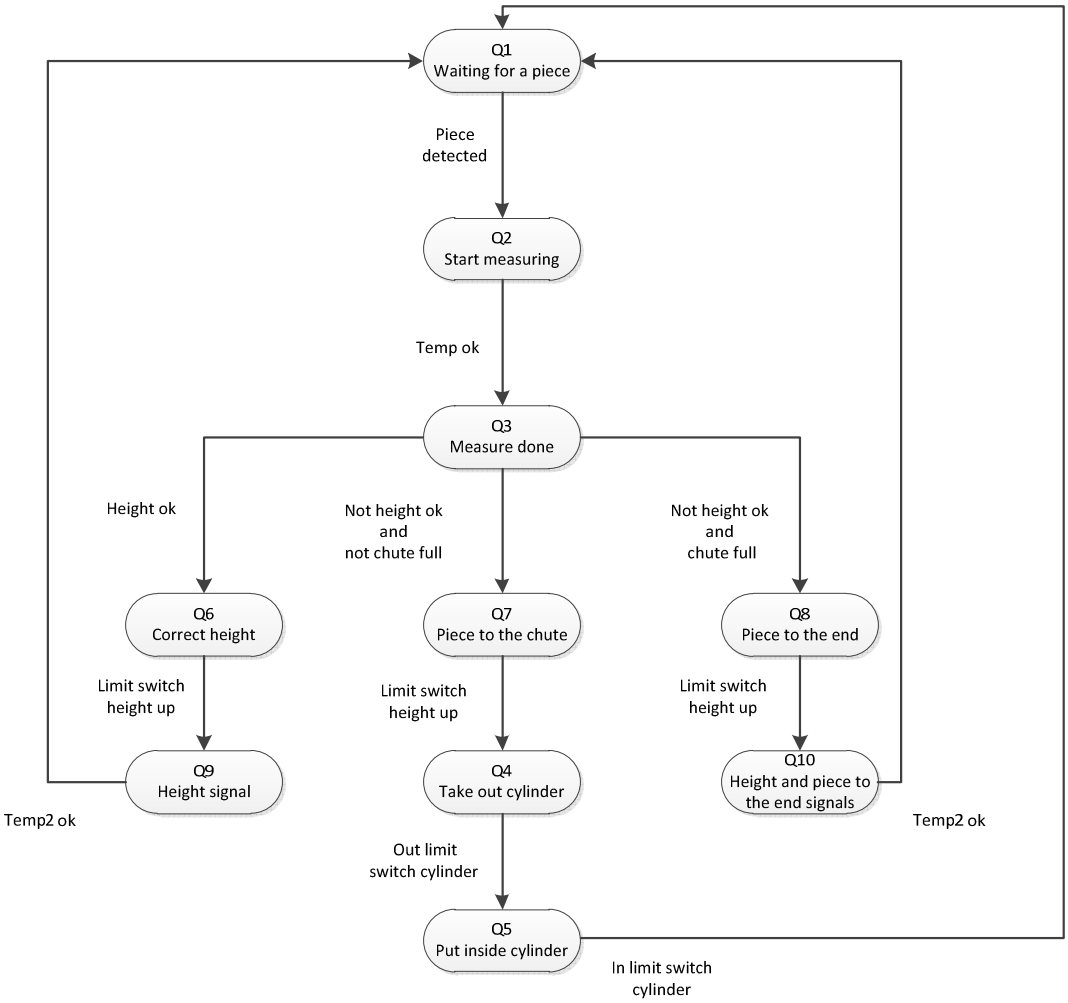
Finally there is the season of black pieces. When a piece reaches the end of the belt, this must stop. Until it isn't removed, it can't load parts.

The right part works in the following way. When a piece reaches the end of the belt and the start button is activated. It proceeds to place the arm in the left position. The arm has to be lowered and extended to take the piece. The arm must be retracted so that it can rotate. When he reaches the right position, the arm must be lowered and extended too. It drops the piece and returns to the starting position.

2.4 Moore state machine

The Moore state diagrams are divided in four. A diagram for the control of the belt; other for the height meter; one for the three stations, that distinguish the kind of the piece, and finally one for the arm of the right part.

2.4.1 Height control diagram



This function controls the measure of height of the pieces. This diagram is implemented in the FB1. It waits for a piece that is detected by the photoelectrical barrier. A piece will be in the correct position and it will move to the next state when it detects a falling edge of the input signal of the position sensor. In the state 2, it

proceeds to send the Stop_height signal and lower the rod that will measure the height of the piece. It activates a timer to be sure that the cylinder has measured correctly.

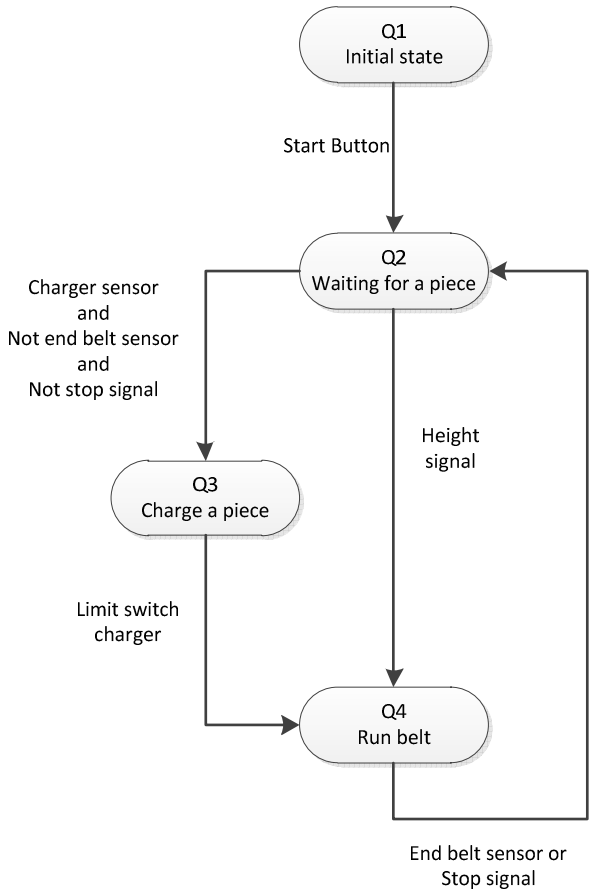
Then I read the memory variable that stores the limit height. To see if the height is correct I have previously saved the value from which the piece is considered correct. It is stored in OB100. And compare it with the value read from the sensor. Depending on the variable and whether the chute is full, there are three options:

The height is correct, in this state it goes down the rod that measures. It waits to come up and sent the belt signal to the FB2 so that it reactivates the belt without loading a new piece. I had to use a timer that counts a second, but because FB2 doesn't read the signal that sent this function.

The second option is that the height is not correct and the chute is not full. To determine whether the chute is full, it uses a variable that is increased each time the program go through the state 7. There is only place for two pieces in the chute. Because of this when the counter is two, the chute will be full. First it has to wait to go up the height rod. After the cylinder is activated so that ejects the piece to the chute. And the cylinder moves back.

The third option is that the measure is correct and the chute is full, so it has to send the piece to the end of the belt. It waits to go up the rod and sends to the belt the signal, but in this case also it actives the piece_to _end signal for this piece is not used for the next three stations.

2.4.2 Belt control diagram



This function controls the operation of the belt that transports the pieces to the different stations. FB2 contains this diagram.

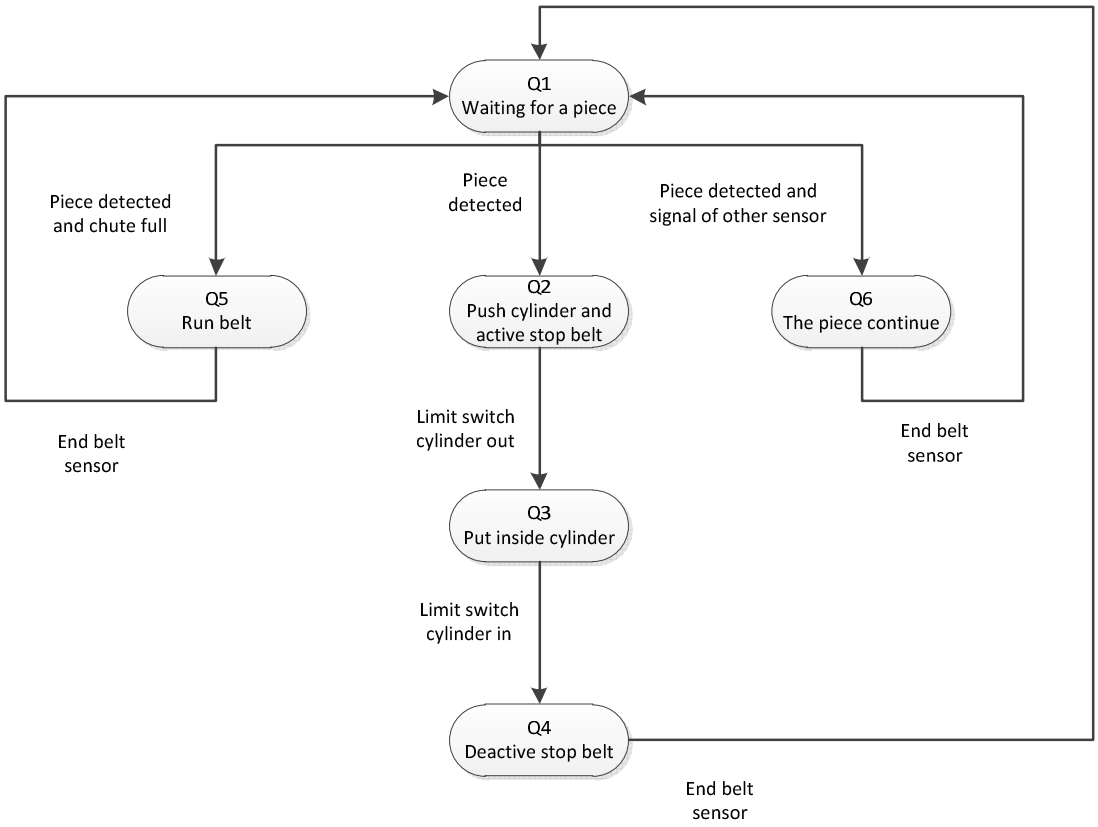
I have chosen like criterion that only one piece can be at once on the belt. This is achieved whit the signals between the different function blocks.

State 2 is activated when you press the start button on the left button panel. And it waits until it finds a piece on the initial stock, there isn't a piece at the end of the belt and stop signals of other stations aren't enabled.

If it meets these requirements, it will be charged a piece on the belt. So the cylinder is activated. It expects that the switch limit is activated and turns on the conveyor belt.

From the state two, there is another option when the height signal is activated. This signal indicates that it has to reactivate the belt and not to be loaded any piece. It goes to the state 4 that waits the signal stop of the other stations or the piece arrives to the end of the belt. This ensures that the piece has gone to the end of the belt or one of the chutes. So there is only a piece on the belt.

2.4.3 Three stations control diagram



It shows the Moore diagram used in the metal white and black stations. It uses the same function because the only thing that varies is the type of sensor. The program is in the FB3.

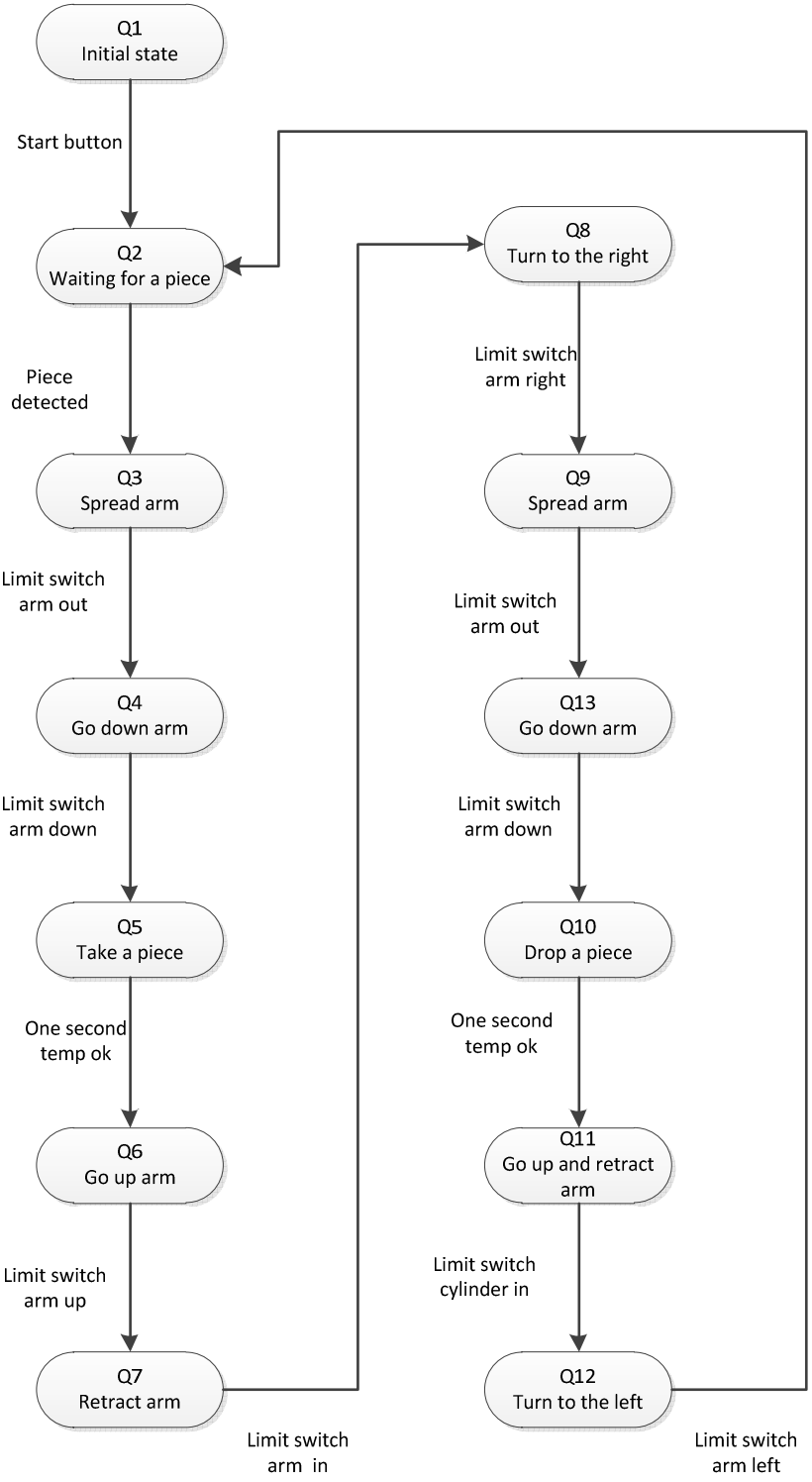
The system is waiting to detect a piece. Then there are three options:

- If there is a part and its own chute is full. It will go to state 5 which reactivates the movement of the belt and run_belt signal is activated. This will indicate to the following stations that the piece has been discarded and they haven't to take it. This is the solution to the problem of the pieces are identified by the following stations. For example the metal parts are detected by the capacitive sensor and the optical. State 5 is turned off by the sensor end of the belt and then it goes to the state 1.

- If another station rejects a piece the signal will be enabled. So it moves to state 6. In this case, nothing is done.

- Finally, if a piece is detected, it is not turned on the signal of another sensor and the own chute is not full. It will go to state 2 in which the cylinder moves the piece and stops the tape, using the external signal Stop_belt. It also increases the counter variable for the number of pieces that are in the chute. The program compares the value of this variable with a constant value 2. When the out limit switch of the cylinder is activated, it will go to the state 3 and the cylinder is retracted. After that the program passes to the state 4 in which the signal stop_belt is turned off.

2.4.4 Arm control diagram



This diagram is in the FB4 and includes the operation of the right part. I've included in a single block function because of its simplicity. Function is activated clicking the start button on the right button panel. The transition from state 2 to 3 is produced by the sensor end of the belt, on the left side of the plant. The next step is to spread the arm.

When it detects that it has reached the position, it will go down the arm. Later it takes the part activating the output signal take_piece.

Then the actions are the following: to raise, to retract and to turn right the arm. The transitions, between the states of these actions, are the respective position sensors. It's necessary to retract and go up the arm, because the layout of the plant does not allow the rotation of the stretched arm.

Once the arm is in position, it shrinks and goes down the arm to approach the piece to the container. It activates the output signal drop_piece. As it isn't loaded, the arm raises and retracts in the same action. Finally, it puts in the original position, rotating to the left the arm.

Chapter 3: Results

3.1The code

The programming has to be fully parametric so that operations are independent of the inputs and outputs. So it can be reused. I have used functions FB / FC.

Considerations to understand the code:

-All code is detailed in the space for comments, I have included comments between the text to all is clear.

-There are some sensors that are normally closed. I have denied this kind of signals in order to all the signals are used in the same way. One example is the input 'Photo_sen_r'.

- Manuals [1] and [2] has been used for implement the code.

3.1.1 OB1

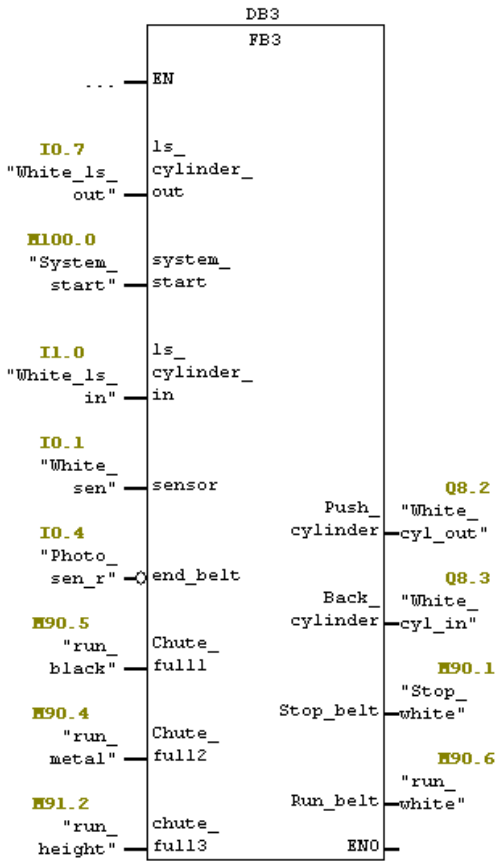
It is the main program code is executed cyclically. It's where it calls all the functions.

OB1 : "Main Program Sweep (Cycle)"

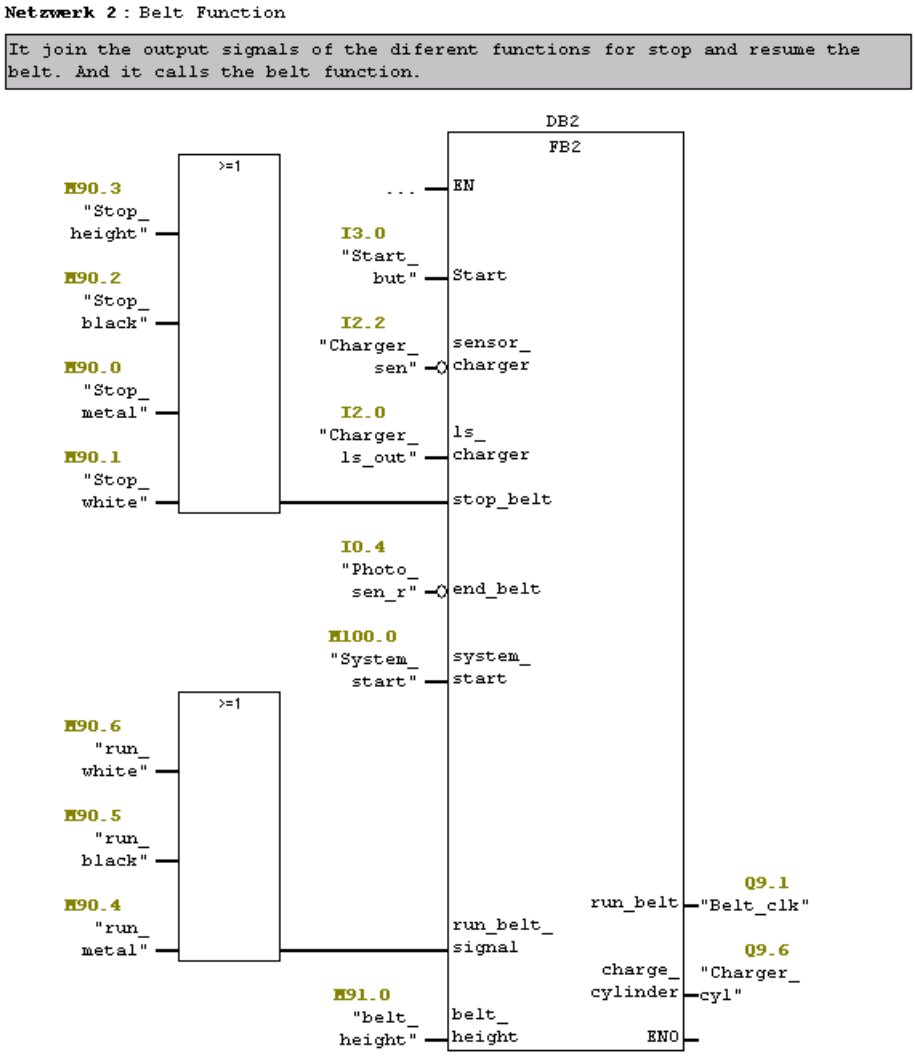
This is the main program that works in a cycle.

Netzwerk 1: Function for the white pieces

It calls to the generic function.



I have to use three different signals for each station on the left part. Stop signals to stop the belt and Run signals to reactivate the movement. Unless I use them, the different function overwritten the same signal and the operation would not be correct. To use a single Stop signal in FB2, I use the OR logic function. For Run signals I can't do the same. Because the three stations don't reactivate the belt and the signal of the height station reactivates the belt.

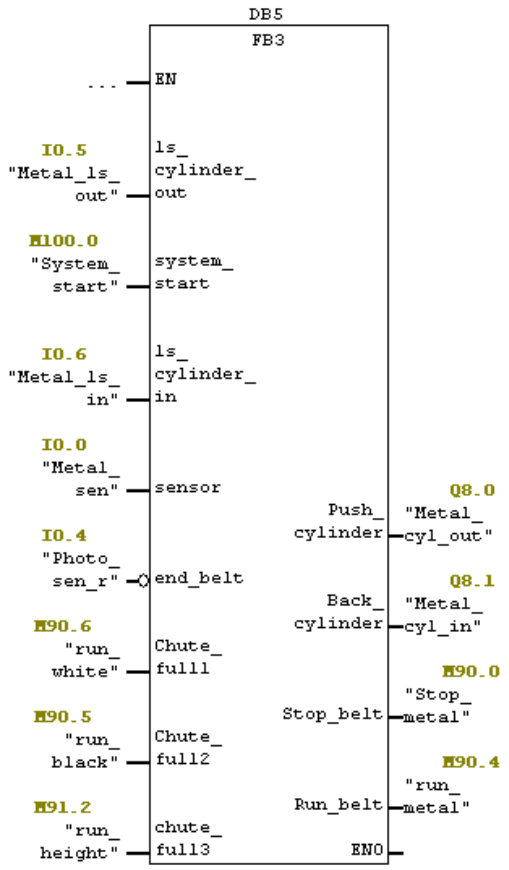


In the following two networks calls the same generic function FB3. Although all have the same operation, they must be stored in different DB because of the data of each station are different.

It connects the Run signals to each block of the other three seasons. So they know when a piece has been discarded by another before station.

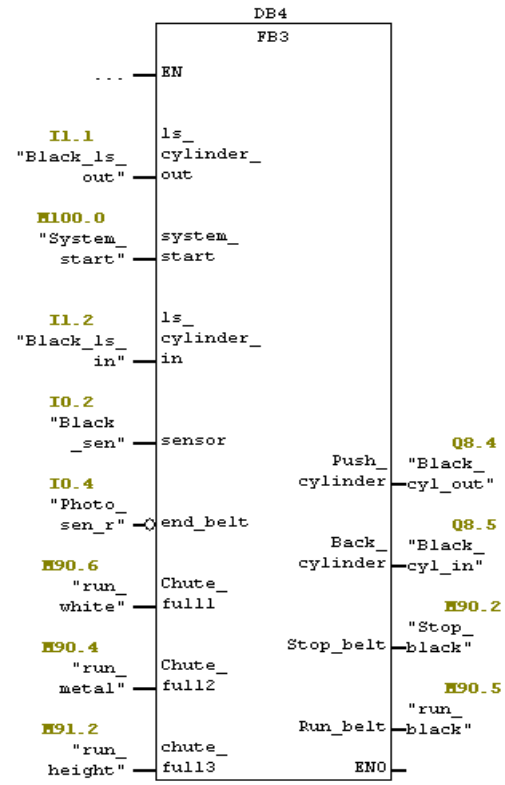
Netzwerk 3: Function for the metal pieces

It calls to the generic function.



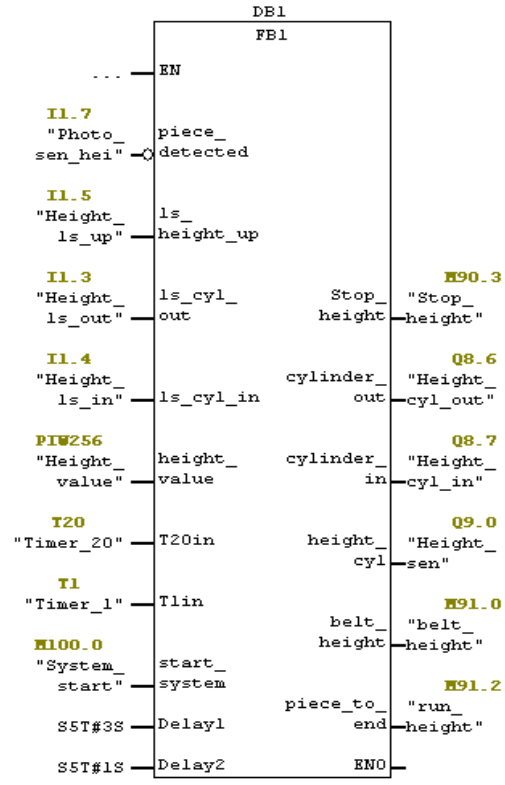
Netzwerk 4 : Function for the black pieces

It calls to the generic function.



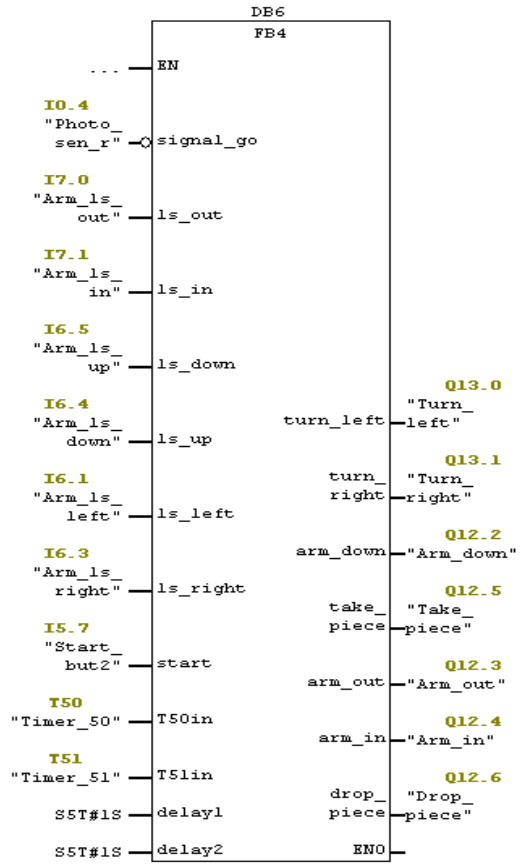
Netzwerk 5 : Function of the height measure

It calls the height measure function.



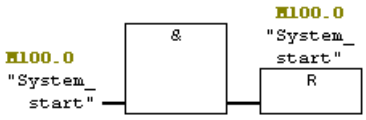
Netzwerk 6 : Function of the right side

It calls the specific function of the right part.



Netzwerk 7 : Reset the System_start variable

It gets the system restart only one time.



This network resets the memory variable ‘System_start’ in order to this is actived the first cycle.

3.1.2 OB 100

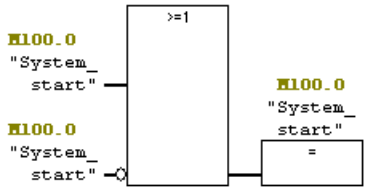
The memory M100.0 is activated only un the first cycle of ejecution.

OB100 : "Complete Restart"

OB100 runs only once cycle and then the OB1.

Netzwerk 1: Set System_start

It sets always the System_start the first cycle.



Netzwerk 2: Define the heighlimit memory

Kommentar:

```
L     l2500
T     "heightlimit"              MW102
```

Heightlimit is a constant used for store the maximum correct value of the height.

3.1.3 FB1

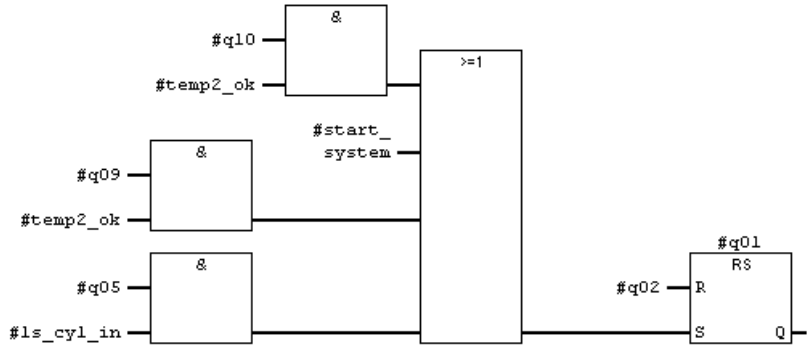
Function block of the height measure station.

FB1 : Height measure Function

This function has all the operations for the height measure.

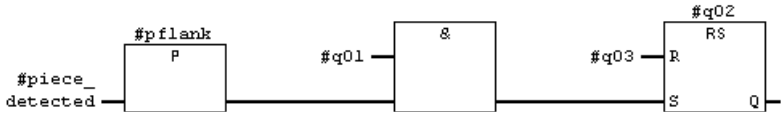
Netzwerk 1: State 1

Waiting for a detected piece.



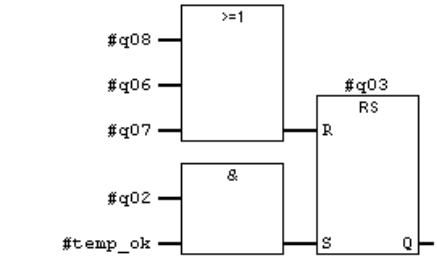
Netzwerk 2: State 2

Piece detected, this state actives a timer.



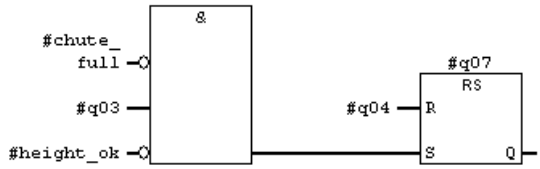
Netzwerk 3 : State 3

Height measure sensor is in the correct position. I can read the height of the piece.



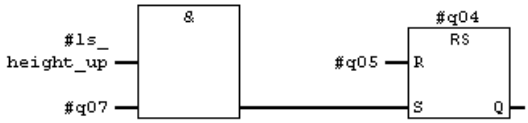
Netzwerk 4 : State 7

The height is correct and the chute isn't full.



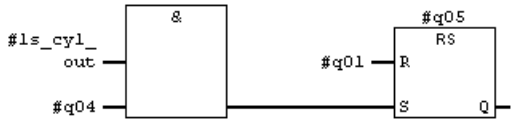
Netzwerk 5 : State 4

It has to wait until the height measure sensor is in the up position. And it pushs the piece to the chute.



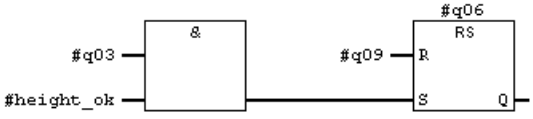
Netzwerk 6 : State 5

Cylinder comes back.



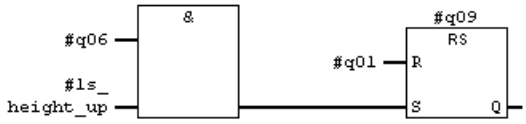
Netzwerk 7 : State 6

The height measure isn't correct.



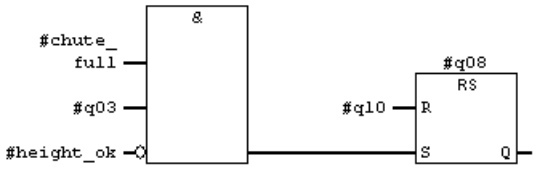
Netzwerk 8 : State 9

It has to wait until the height measure sensor is in the up position. And it pushs the piece to end of the belt.



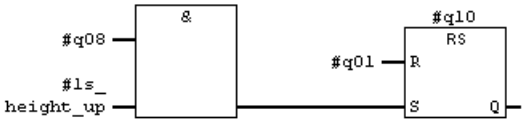
Netzwerk 9 : State 8

The height is correct and the chute is full.



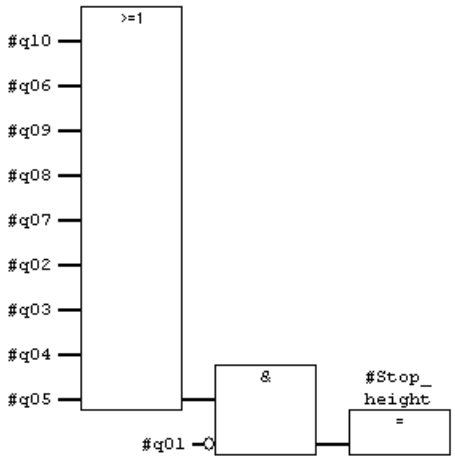
Netzwerk 10 : State 10

It has to wait until the height measure sensor is in the up position. And it pushs the piece to end of the belt.



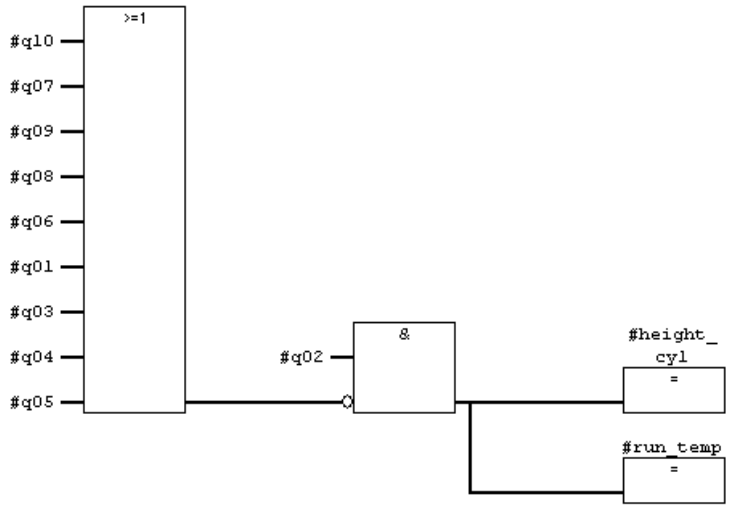
Netzwerk 11 : Titel:

Stop_Height is a signal for the function block2 knows that there is a piece in the height measure detector. And it can´t move the belt.



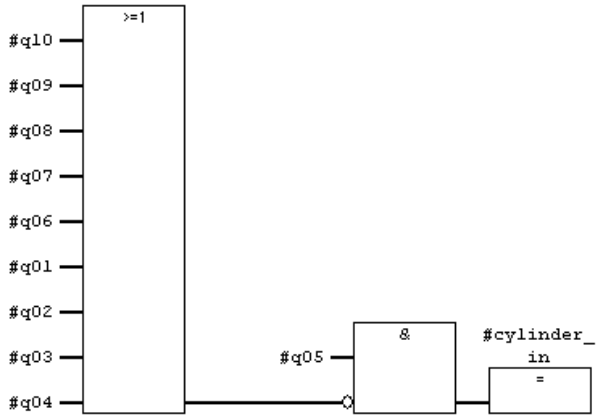
Netzwerk 12 : Titel:

The state 2 actives the temp for wait until the sensor is in the correct position. And down the height measure sensor



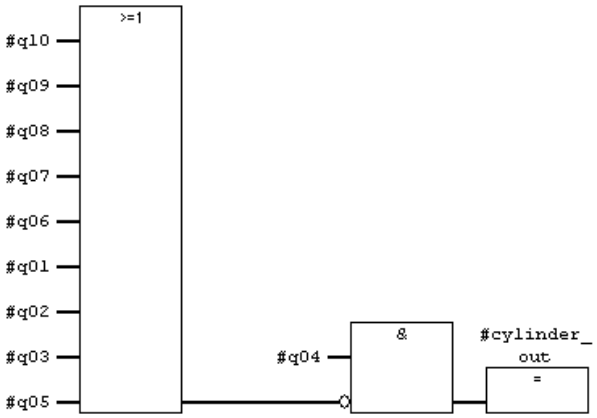
Netzwerk 13 : Titel:

Go back the charger cylinder.



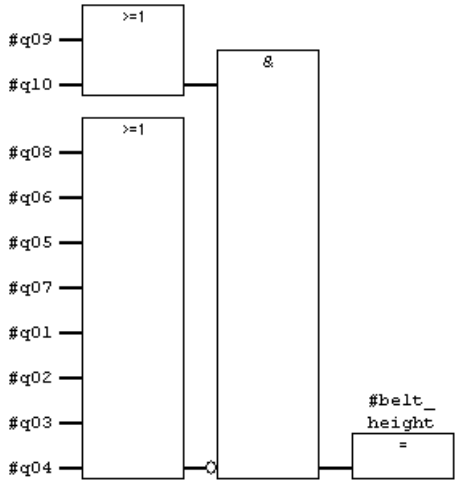
Netzwerk 14 : Titel:

Take out the charger cylinder.



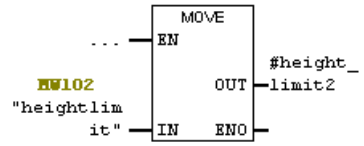
Netzwerk 15 : Titel:

Signal for resume the belt movement.



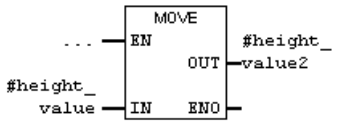
Netzwerk 16 : Titel:

It changes the format of the variable height_limit, it´s the maximum value for considerate correct the heiht value of the piece.



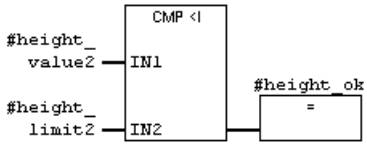
Netzwerk 17 : Titel:

It changes the format of the variable height_value, it´s the real value of the height measure.



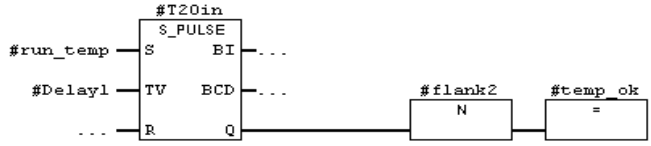
Netzwerk 18 : Titel:

Compare the real height measure with a constant for know if the piece is in the correct position.



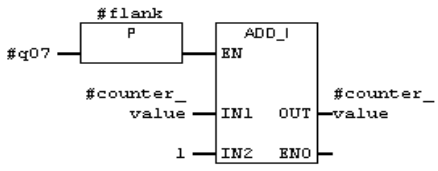
Netzwerk 19 : Titel:

Timer counts three seconds



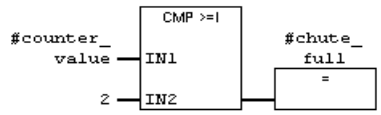
Netzwerk 20 : Titel:

It counts the piece that there is in the height chute.



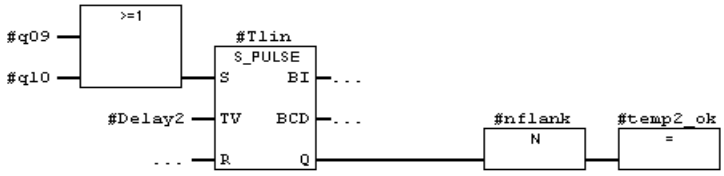
Netzwerk 21 : Titel:

Two is maximum pieces in the chute.



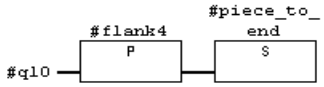
Netzwerk 22 : Titel:

I need to have actived the output signal belt_height one second. If I don't do this the FB2 don't detected this signal. The states 9 and 10 activates the timer.



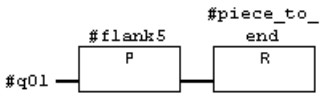
Netzwerk 23 : Titel:

This output signal is for the belt function, it indicates that the piece has to go to the end of the belt.It can't be detected for the others functions.



Netzwerk 24 : Titel:

Reset the output signal.



Definition of the local variables.

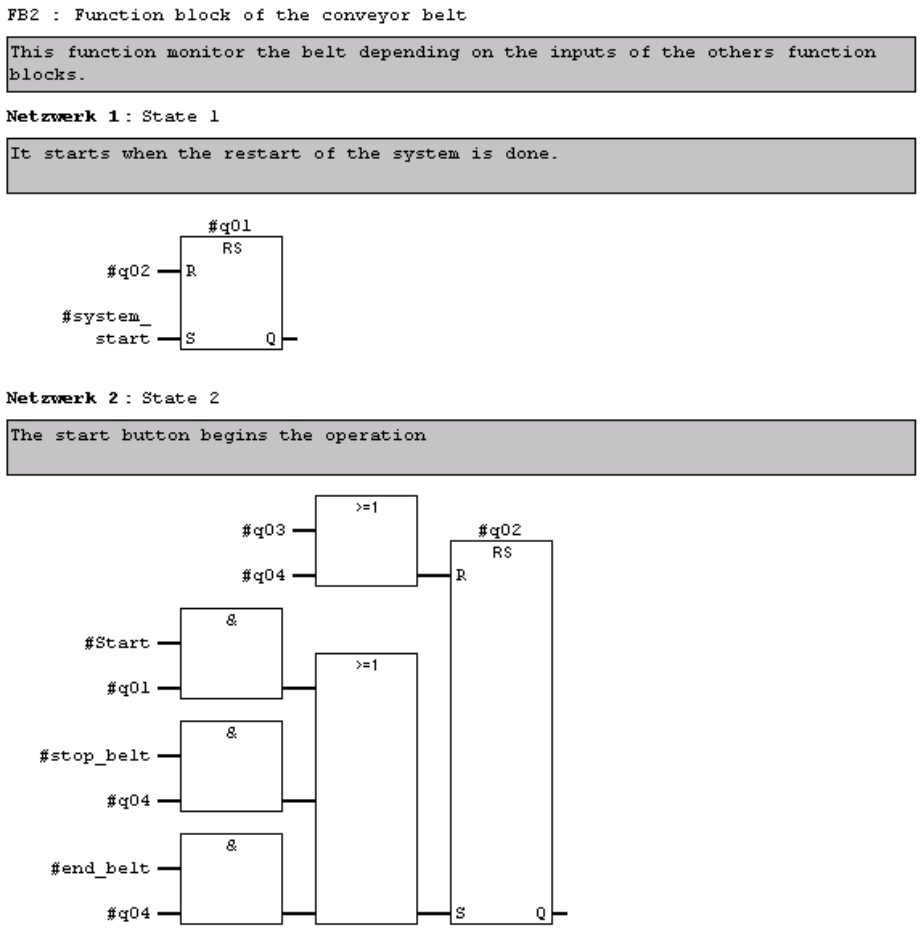
Inputs		Outputs		Static	
Name	Type	Name	Type	Name	Type
piece_detected	Bool	Stop_height	Bool	q01	Bool
ls_height_up	Bool	cylinder_out	Bool	q02	Bool
ls_cyl_out	Bool	cylinder_in	Bool	q03	Bool
ls_cyl_in	Bool	height_cyl	Bool	q04	Bool
height_value	Word	belt_height	Bool	q05	Bool
start_system	Bool	piece_to_end	Bool	q06	Bool
T20in	Timer			q07	Bool
T1in	Timer			q08	Bool
Delay1	S5Time			q09	Bool
Delay2	S5Time			q10	Bool
				height_value2	Integer
				height_limit2	Integer
				counter_value	Integer
				run_temp	Bool
				chute_full	Bool

	temp_ok	Bool
	temp2_ok	Bool
	pflank	Bool
	flank2	Bool
	Flank	Bool
	nflank	Bool
	flank4	Bool
	flank5	Bool

Table 3: *Local variables of the FB1.*

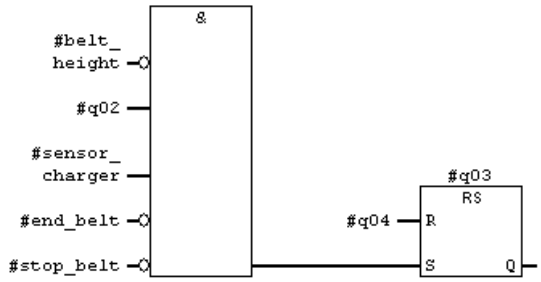
3.1.4 FB2

Function block of the conveyor belt.



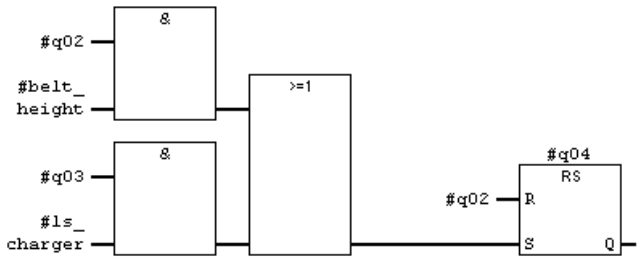
Netzwerk 3 : State 3

It charges a new piece if there's in the charger.



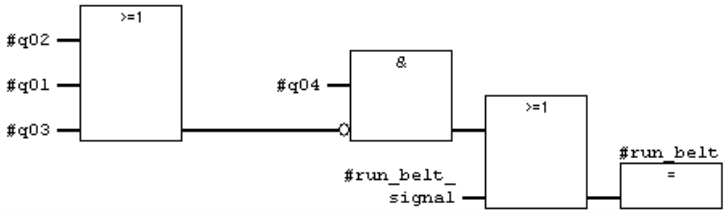
Netzwerk 4 : State 4

It moves the belt. If belt height signal is activated, it means that it don´t charge a new piece. Only it restarts the belt movement because the FB1 orders it.



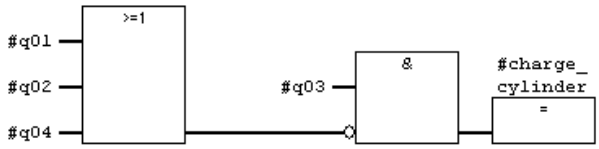
Netzwerk 5 : Titel:

State 4 or the output signal active the movement belt. Run_belt_signal come from the metal, white and black function blocks.



Netzwerk 6 : Titel:

State 3 actives the piece charger.



Definition of the local variables.

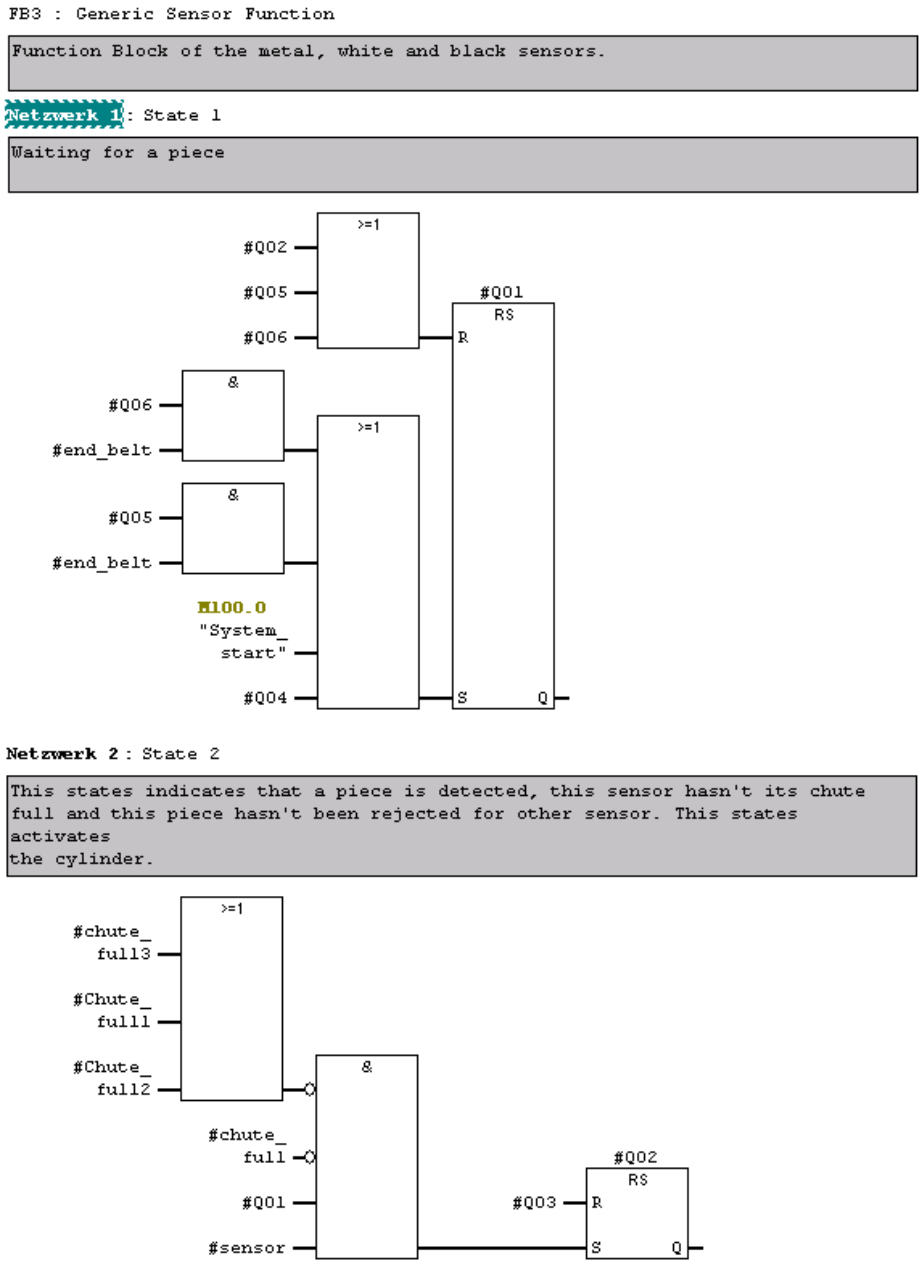
Input		Output		Static	
Name	Type	Name	Type	Name	Type
Start	Bool	run_belt	Bool	q01	Bool
sensor_charger	Bool	charge_cylinder	Bool	q02	Bool
ls_charger	Bool			q03	Bool
stop_belt	Bool			q04	Bool

end_belt	Bool	q05	Bool
system_start	Bool		
run_belt_signal	Bool		
belt_height	Bool		

Table 4: *Local variables of the FB2.*

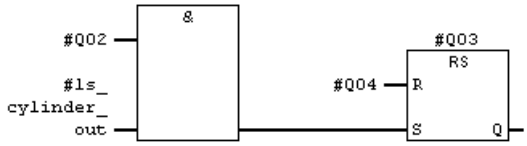
3.1.5 FB3

Function block of the metal, white and black piece station.



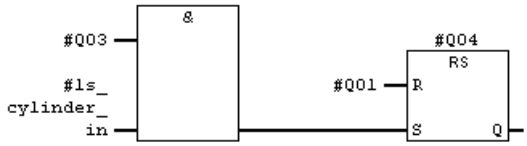
Netzwerk 3 : State 3

The cyilinder comes back.



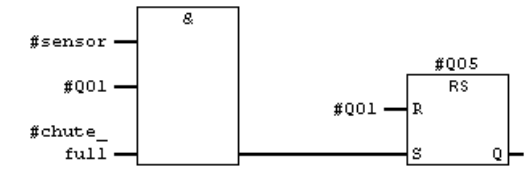
Netzwerk 4 : State 4

It resets the signal stop_belt



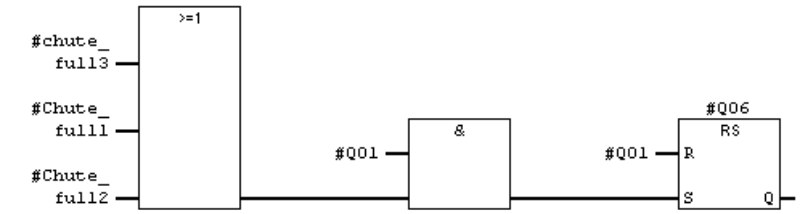
Netzwerk 5 : State 5

The chute of the sensor is full



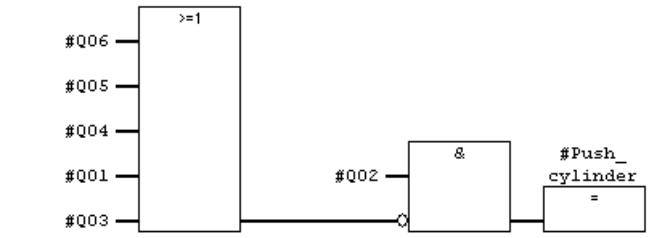
Netzwerk 6 : State 6

This piece haven´t be detect because it comes of other sensor. Because the othere sensor has its chute full.



Netzwerk 7 : Titel:

Push the sensor cylinder



Netzwerk 8 : Titel:

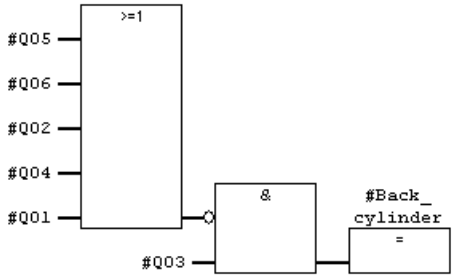
It activates the signal stop_belt for the belt function block.

Netzwerk 9 : Titel:

It deactivates the signal stop_belt for the belt function block.

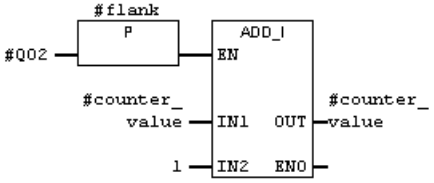
Netzwerk 10 : Titel:

It goes back sensor cylinder



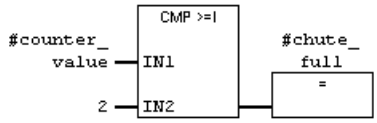
Netzwerk 11 : Titel:

Counter for the chute.



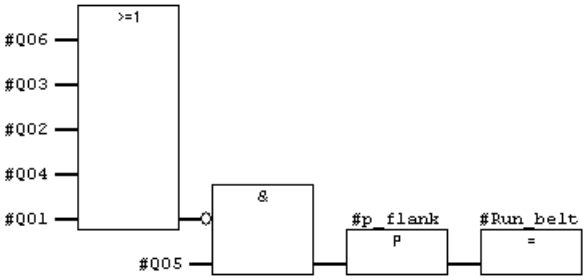
Netzwerk 12 : Titel:

When the counter value is 2, the chute is full



Netzwerk 13 : Titel:

This signal is for the belt function block.



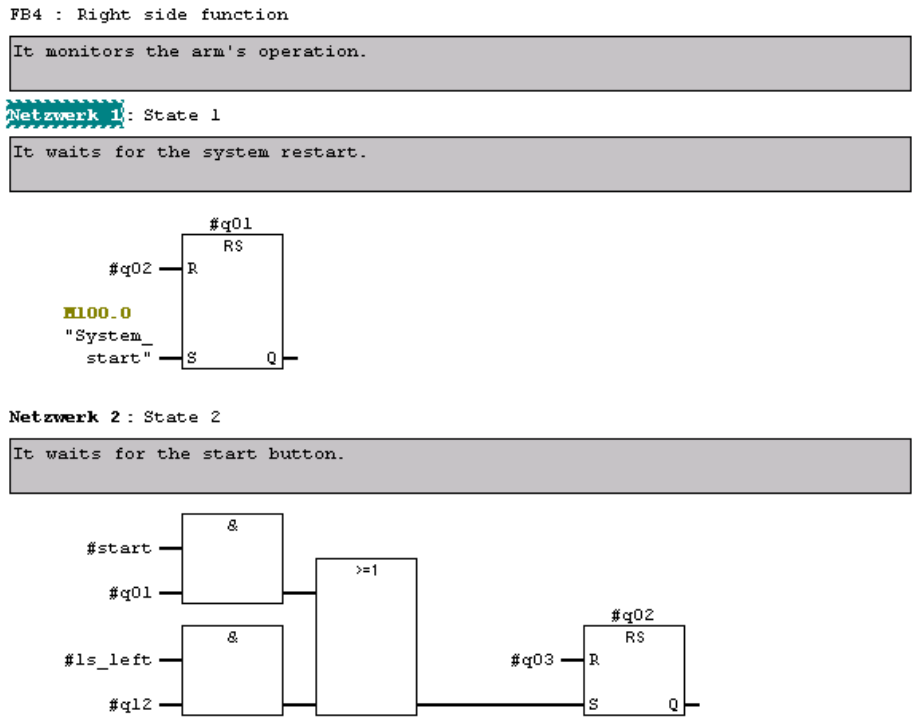
Definition of the local variables.

Input		Output		Static	
Name	Type	Name	Type	Name	Type
ls_cylinder_out	Bool	Push_cylinder	Bool	q01	Bool
system_start	Bool	Back_cylinder	Bool	q02	Bool
ls_cylinder_in	Bool	Stop_belt	Bool	q03	Bool
Sensor	Bool	Run_belt	Bool	q04	Bool
end_belt	Bool			q05	Bool
Chute_full1	Bool			q06	Bool
Chute_full2	Bool			value_2	Integer
chute_full3	Bool			counter_value	Integer
				chute_full	Bool
				p_stop	Bool
				n_stop	Bool
				Flank	Bool
				p_flank	Bool

Table 5: *Local variables of the FB3.*

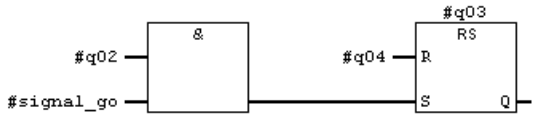
3.1.6 FB4

Function block of the robotic arm.



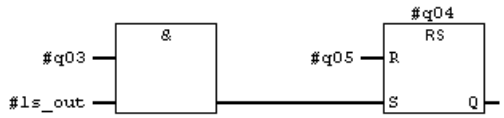
Netzwerk 3 : State 3

There is a piece in the end of the belt.



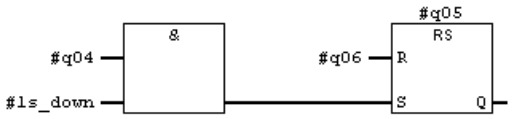
Netzwerk 4 : State 4

The arm is out.



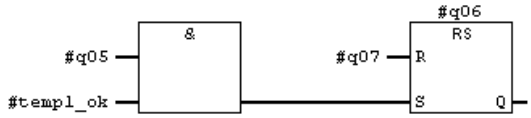
Netzwerk 5 : State 5

The arm is out and down.



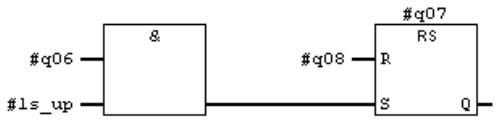
Netzwerk 6 : State 6

It has to wait one seconds for be sure that the arm takes the piece.



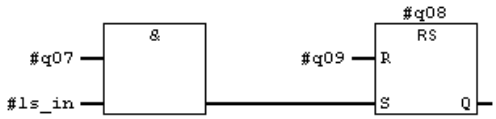
Netzwerk 7 : State 7

The arm is out and up.



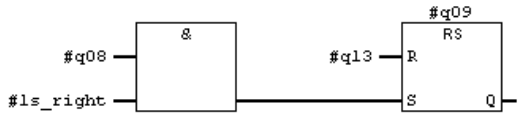
Netzwerk 8 : State 8

The arm is in and up.



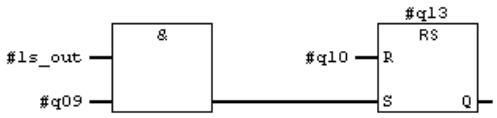
Netzwerk 9 : State 9

The arm has turned to the right position.



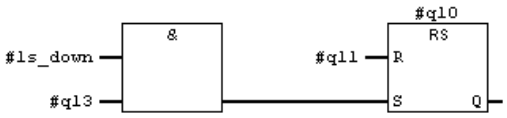
Netzwerk 10 : State 13

The arm is the out and up position.



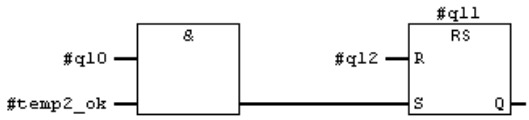
Netzwerk 11 : State 10

The arm is out and down.



Netzwerk 12 : State 11

It has to wait one seconds for be sure that arm lets go the piece.

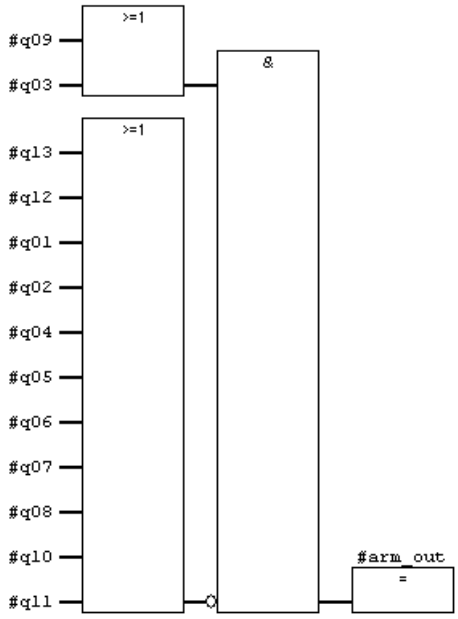


Netzwerk 13 : State 12

The arm is up and in.

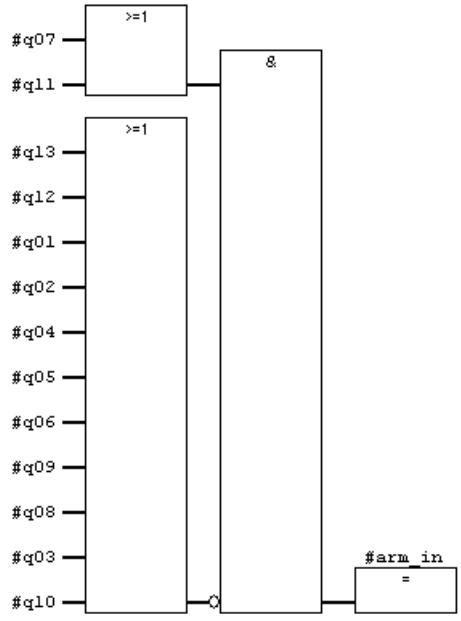
Netzwerk 14 : Titel:

These states take the arm out.



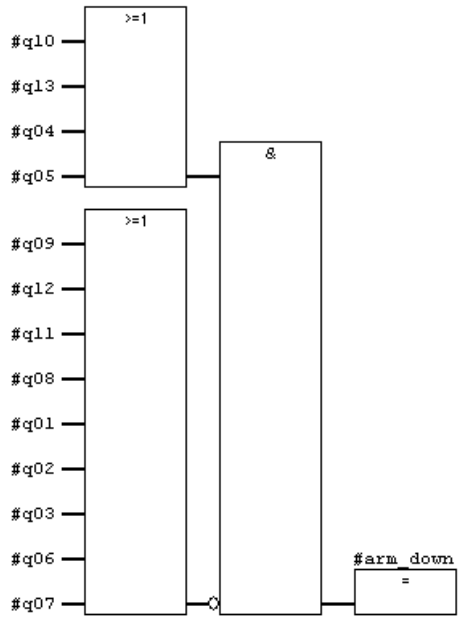
Netzwerk 15 : Titel:

These states take the arm in.



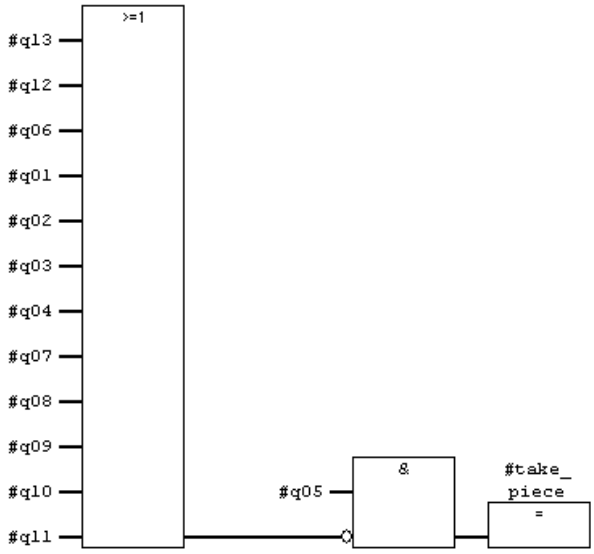
Netzwerk 16 : Titel:

The arm goes down. When the states doesn't activate the output, the arm is in the lower position.



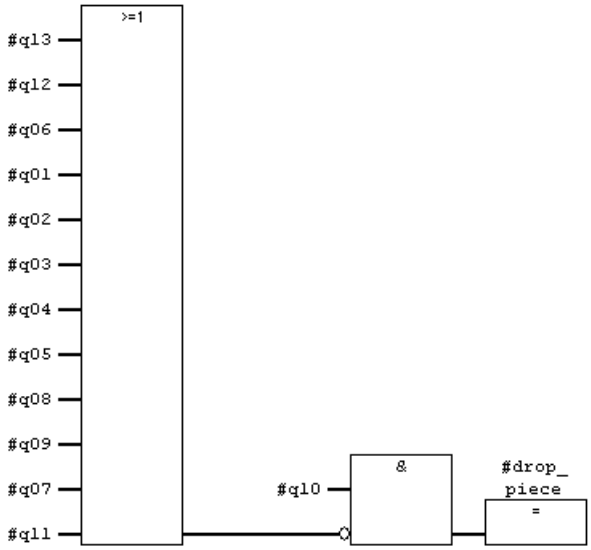
Netzwerk 17 : Titel:

The arm takes the piece.



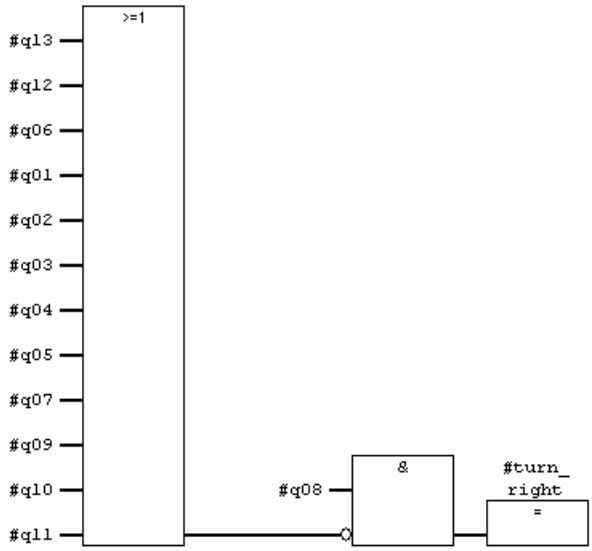
Netzwerk 18 : Titel:

The arm drops the piece.



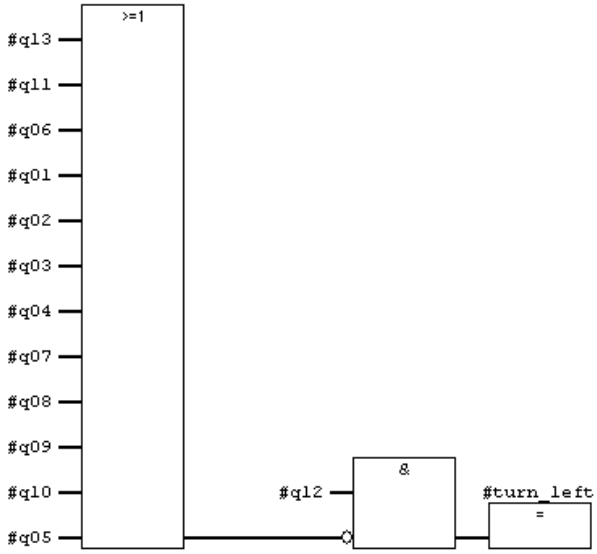
Netzwerk 19 : Titel:

The arm turns to the right.



Netzwerk 20 : Titel:

The arm turns to the left.

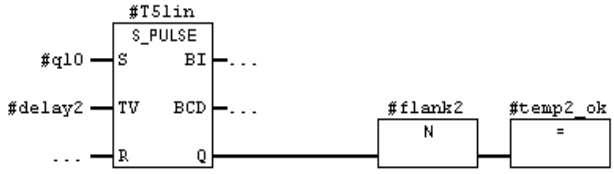


Netzwerk 21 : Titel:

Timer for wait a second while the arm takes the piece.

Netzwerk 22 : Titel:

Timer for wait a second while the arm drops the piece.



Definition of the local variables.

Input		Output		Static	
Name	Type	Name	Type	Name	Type
signal_go	Bool	turn_left	Bool	q01	Bool
ls_out	Bool	turn_right	Bool	q02	Bool
ls_in	Bool	arm_down	Bool	q03	Bool
ls_down	Bool	take_piece	Bool	q04	Bool
ls_up	Bool	arm_out	Bool	q05	Bool
ls_left	Bool	arm_in	Bool	q06	Bool
ls_right	Bool	drop_piece	Bool	q07	Bool

Start	Bool	q08	Bool
T50in	Timer	q09	Bool
T51in	Timer	q10	Bool
delay1	S5Time	q11	Bool
Delay2	S5Time	q12	Bool
		q13	Bool
		temp1_ok	Bool
		temp2_ok	Bool
		flank	Bool
		flank2	Bool

Table 6: *Local variables of the FB4.*

3.2 Global variables table

The following table describes all global inputs and outputs variables used in the code:

Name	Address		Description
Metal_sen	I	0.0	Metal sensor
White_sen	I	0.1	White sensor
Black _sen	I	0.2	Black sensor
Photo_sen_l	I	0.3	Photoelectric sensor left
Photo_sen_r	I	0.4	Photoelectric sensor right
Metal_ls_out	I	0.5	Out limit switch of the metal cylinder
Metal_ls_in	I	0.6	In limit switch of the metal cylinder
White_ls_out	I	0.7	Out limit switch of the white cylinder
White_ls_in	I	1.0	In limit switch of the white cylinder
Black_ls_out	I	1.1	Out limit switch of the black cylinder
Black_ls_in	I	1.2	In limit switch of the black cylinder
Height_ls_out	I	1.3	Out limit switch of the height cylinder
Height_ls_in	I	1.4	In limit switch of the height cylinder
Height_ls_up	I	1.5	Up limit switch of the height cylinder
Height_ls_down	I	1.6	Down limit switch of the height cylinder
Photo_sen_hei	I	1.7	Photoelectric sensor before height sensor
Charger_ls_out	I	2.0	Out limit switch of the charger cylinder
Charger_ls_in	I	2.1	In limit switch of the charger cylinder
Charger_sen	I	2.2	Charger sensor
Start_but	I	3.0	Start button of the left side
Start_but2	I	5.7	Start button of the right side
Arm_ls_left	I	6.1	Left sensor of the arm
Arm_ls_right	I	6.3	Right sensor of the arm
Arm_ls_down	I	6.4	Down limit switch of the arm
Arm_ls_up	I	6.5	Up limit switch of the arm
Arm_ls_out	I	7.0	Out limit switch of the arm

Arm_ls_in	I	7.1	In limit switch of the arm
Metal_cyl_out	Q	8.0	Metal cylinder out
Metal_cyl_in	Q	8.1	Metal cylinder in
White_cyl_out	Q	8.2	White cylinder out
White_cyl_in	Q	8.3	White cylinder in
Black_cyl_out	Q	8.4	Black cylinder out
Black_cyl_in	Q	8.5	Black cylinder in
Height_cyl_out	Q	8.6	Height cylinder out
Height_cyl_in	Q	8.7	Height cylinder in
Height_sen	Q	9.0	Sensor height
Belt_clk	Q	9.1	Belt clockwise
Start_lig	Q	9.2	Start light
Stop_lig	Q	9.3	Stop light
Quit_lig	Q	9.4	Quittierung light
Charger_cyl	Q	9.6	Charger cylinder
Belt_anticlk	Q	9.7	Belt anticlockwise
Arm_down	Q	12.2	Arm cylinder down
Arm_out	Q	12.3	Arm cylinder out
Arm_in	Q	12.4	Arm cylinder in
Take_piece	Q	12.5	Arm takes a piece
Drop_piece	Q	12.6	Arm drops a piece
Turn_left	Q	13.0	Arm turn to the left
Turn_right	Q	13.1	Arm turn to the right

Table 7: *Used variables of the program.*

Chapter 4: Future improvements

Some possible future improvements are:

-Modify the program so it can have multiple pieces on the belt. So it would get a greater amount of pieces treated in a shorter time.

- It would be possible to use fewer signals between Function Blocks.

-In the generic functions, I should connect only the signals of the previous stations not all.

- It can be implemented more functions for the system like system downtime, emergencies stops and operate the plant with the switches and buttons of the plant.

LIST OF REFERENCES

[1] *Labor Industrielle Steuerungen*. Prof. Dr.-Ing. M. Haas. Ostfalia Hochschule.

http://www.ostfalia.de/export/sites/default/de/pws/haas/Lehre/Labore/LaborSPS/Laborversuche.pdf

[2] *SIMATIC. Programming with STEP 7. Manual.* Siemens AG. Edition 03/2006

http://cache.automation.siemens.com/dnl/zU/zUwMjUxNTMA_18652056_HB/S7prv54_e.pdf